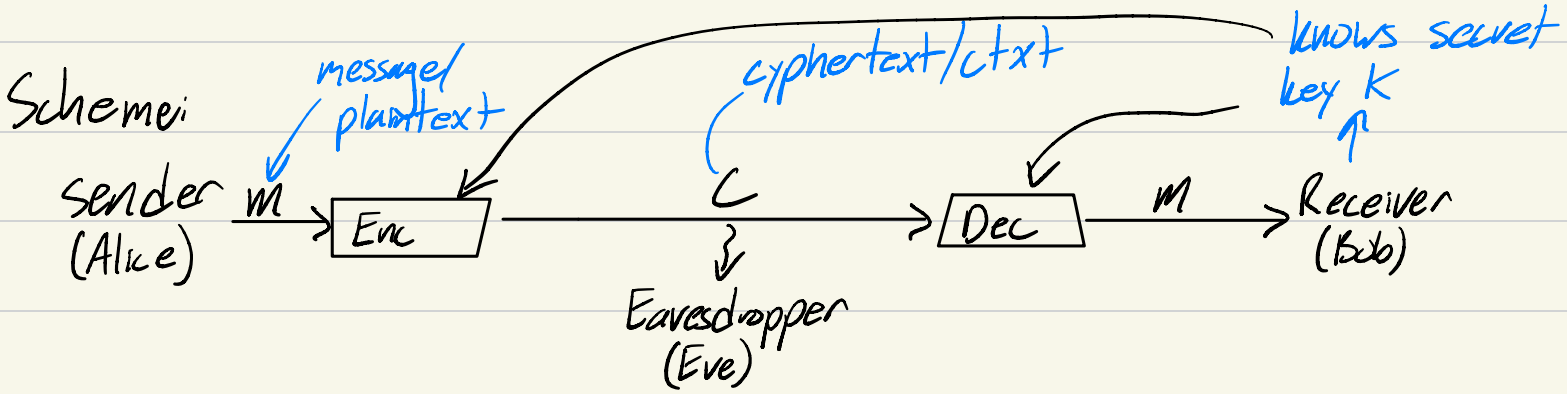
1A - 1/6

Kerckhoffs principle: encryption scheme shouldn't be assumed to be secret

- Shannon turned crypto from art to science
  - ↳ also started to be important to the public in the 70s



- in public key cryptography, Alice uses a public key that Eve can see, yet Alice can still communicate w/ Bob securely

Shift Cypher

M: any length string of alphabetic chars
  - ↳ $M \in \{A, \ldots, Z\}^*$

K: shift, $\in \{0, \ldots, 25\}$

$C = Enc(K, M)$: shift each char in M by K places

Problem: only 26 keys! can brute force
  - ↳ **lesson 1: need lots of keys**

- why Kerckhoff?
  - ↳ need fewer schemes, open source benefits, etc.

↳ so we assume everything a/b the scheme is public except the key

New Approach: Vigenère Cipher

$K = (k_1, \ldots, k_{10})$, $k_i \in [25]$

$M = (m_1 \; m_2 \; \ldots \; m_{10} \; m_{11} \; \ldots)$

k_1 shifts k_2 ... k_{10} k_1 shifts

$C = (c_1 \; c_2 \; \ldots)$

- How did we break it?

↳ well c_1, c_{11}, c_{21}, etc. all had the same key

↳ so we can perform frequency analysis on the ctxt; e.g. most common letter will be "e", etc. Can cycle thru lengths of K

One-Time Pad (OTP)

$M \in \{0,1\}^n$

(1.2.1) Construction

For $n \geq 1$, $M, K \in \{0,1\}^n$, $Enc(K, M) = K \oplus M$ (XOR) → choose K uniformly ~ $\{0,1\}^n$

For $C, K \in \{0,1\}^n$, $Dec(K, M) = K \oplus M$

Claim: $\forall n \geq 1$, $M, K \in \{0,1\}^n$, $Dec(Enc(K, M)) = M$

Pf: $Dec(Enc(K, M)) = K \oplus (K \oplus M)$

$= (K \oplus K) \oplus M) = 0 \oplus M = M$

Sections 1.3–1.6, 2.1–2.3

1B - 1/8

OTP

Def. For $n \geq 1$, $M, K, C \in \{0,1\}^n$: $(K \xleftarrow{\$} \{0,1\}^n)$

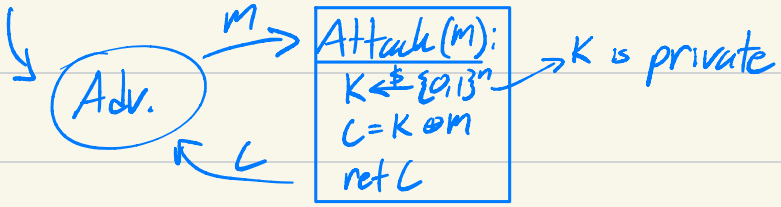
$Enc(K, M)$: ret $C := K \oplus M$, $Dec(K, M)$: ret $M := K \oplus C$

Notation. $X \xleftarrow{\$} S \equiv X \twoheadleftarrow S$: RV X sampled uniformly from S.

- we want to make a security claim: if the victims do ① (what they're supposed to do), and the attacker does ② (capabilities), then ⟨some security property⟩ ⇝ "attack scenario"

① $K \xleftarrow{\$} \{0,1\}^n$

② we assume the adv. can see C & influence M!



- to prove security, we want C to look like a uniformly random string to the adversary.

Claim (1.4.1). $\forall n \geq 1$, $\forall M \in \{0,1\}^n$, the output of $Attack(M)$ is uniform over $\{0,1\}^n$.

Pf.

Fix $n \geq 1$, $M \in \{0,1\}^n$. WTS: for any $C \in \{0,1\}^n$, $P[Attack(M) = C] = 1/2^n$

Fix $C \in \{0,1\}^n$.

$$P[Attack(M) = C] = P[K \oplus M = C]$$
$$= P[K \oplus M \oplus M = C \oplus M]$$
$$= P[K = C \oplus M] = \frac{1}{2^n}$$ bc K picked U @ rand., C & M fixed

- the adv. thus can't tell the diff between Attack(M) and a random $C \xleftarrow{\$} \{0,1\}^n$.
- but the OTP breaks if the same key is used multiple times

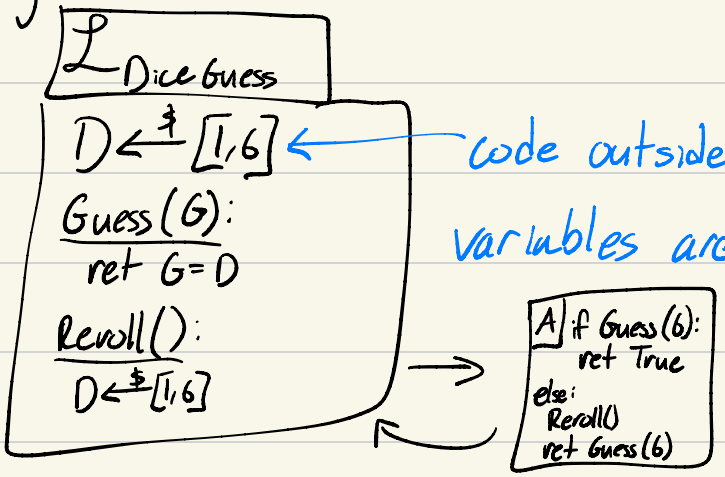
Machinery

2.1.1

Def: Library. A collection of subroutines w/ an interface.

Def: Interface. All subroutines in a library w/ their input & output types.

E.g.



code outside fn.s ran first

variables are global to library, private to external

Def: Linking. $A \diamond L \Rightarrow$ output x; alg. A makes subroutine calls on L, w/ output x.

↳ e.g. above, $P[\overline{A \diamond L} \Rightarrow \text{True}] = 1 - \frac{5}{6} \cdot \frac{5}{6} = \frac{11}{36}$

- define $\mathcal{L}_{OTP}$:

```
Attack(m):
K ←$ {0,1}^n
C = K ⊕ m
ret C
```

$\mathcal{L}_{RAND}$:

```
Attack(m):
C ←$ {0,1}^n
ret C
```

Def: Interchangeable. $\mathcal{L}_1$ and $\mathcal{L}_2$ are interchangeable (denoted $\mathcal{L}_1 \equiv \mathcal{L}_2$) if $\forall$ calling algs. A w/ bool. output,

$$P[\overline{A \diamond \mathcal{L}_1} \Rightarrow \text{True}] = P[\overline{A \diamond \mathcal{L}_2} \Rightarrow \text{True}]$$

Claim 1.4.1 (rewritten). $\mathcal{L}_{OTP} \equiv \mathcal{L}_{RAND}$

↳ does this hold if we change bitwise XOR to bitwise AND?

↳ no. How? find A s.t. $P[\overline{A \diamond \mathcal{L}_1} \Rightarrow T] \neq P[\overline{A \diamond \mathcal{L}_2} \Rightarrow T]$, and remember that the adv. controls the message

```
A:
M = 0^n
C = L.Attack(M)
if C = 0^n: ret True
else: ret False
```

$P[\overline{A \diamond \mathcal{L}_1} \Rightarrow T] = 1$

$P[\overline{A \diamond \mathcal{L}_2} \Rightarrow T] = \frac{1}{2^n}$ ≠!

2A - 1/13

Proving Interchangeability

+ "Hybrid technique" (interchangeability is transitive)
  ↳ $\mathcal{L}_1 \equiv \mathcal{L}_2 \equiv \dots \equiv \mathcal{L}_{end}$
    ↳ ideally any one of these equivalences is fairly trivial

Chain Rule. If $\mathcal{L}_1 \equiv \mathcal{L}_2$, $\mathcal{L}^* \diamond \mathcal{L}_1 \equiv \mathcal{L}^* \diamond \mathcal{L}_2$ $\forall \mathcal{L}^*$.

Pf.

Fix A. $P[A \diamond (L^* \diamond L_1) \Rightarrow T] = P[(A \diamond L^*) \diamond L_1 \Rightarrow T]$

this must be $\in \{\mathcal{L}^*\}$!

$= P[(A \diamond L^*) \diamond L_2 \Rightarrow T]$ ▨

Claim 2.9.2. The following libraries are interchangeable (both ret X, Y s.t. $X \oplus Y = M$, $n \geq 1$):

L_1

```
Samp(M):
X ←$ {0,1}^n
Y = X ⊕ M
ret (X, Y)
```

$\equiv$

```
Samp(M):
Y ←$ {0,1}^n
X = Y ⊕ M
ret (X, Y)
```

L_2

Pf.

① L_1

② $L_2 =$

```
Samp(M):
"   "
  ⋮
X' = Y ⊕ M
ret (X, Y)
```

→ $X = X' = Y \oplus M$ always

③ L_3 =
```
Samp(m):
x ←$ {0,1}^n
Y ← x ⊕ m
X' ← Y ⊕ m
ret (X', Y)
```
→ same as OTP!

④ L_4 =
```
Samp(m):
Y = Enc(m)
X' = Y ⊕ m
ret (X', Y)
```
◇ L_{OTP}

~~A~~ this proof basically just says if $X \sim U(\{0,1\}^n)$, $Y = X \oplus M$, then Y is unif. over $\{0,1\}^n$.

⑤ Replace L_{OTP} w/ $L_{OTP\text{-}Rand}$ (by chain rule)

⑥ L_6 =
```
Samp(m):
Y ←$ {0,1}^n
X' = Y ⊕ m
ret (X', Y)
```

⑦ L_7 =
```
Samp(m):
Y ←$ {0,1}^n
X = Y ⊕ m
ret (X, Y)
```
▨

Def: Three-hop Maneuver. $L_1 \equiv L_2 \diamond L_A \equiv L_2 \diamond L_B \equiv L_3$.

Defining Primitives

- must define syntax, correctness, security

Schemes denoted Σ

Def: Symmetric/Secret Key Encryption.

① Syntax: keyspace $\mathcal{K}$, message space $\mathcal{M}$, ctxt space $\mathcal{C}$, $Enc: \mathcal{K} \times \mathcal{M} \to \mathcal{C}$ (random), $Dec: \mathcal{K} \times \mathcal{C} \to \mathcal{M}$ (nonrandom)

② Correctness: $\forall K \in \mathcal{K}, m \in \mathcal{M}, P[Dec(K, Enc(K,m)) = m] = 1$

③ **Def: One-Time Secrecy.** SKE scheme Σ has "OTS" if the following libraries are interchangeable:

L_{OTS}:
```
Enc(m):
K ←$ Σ.K
C = Σ.Enc(K,m)
ret C
```

$L_{OTS\text{-}Rand}$:
```
Enc(m):
C ←$ Σ.C
ret C
```

- SKE is much faster than PKE

+ Idea: $\Sigma_1, \Sigma_2 \to \Sigma^*$

$Enc^*(K_1, M)$: ???

$K_2 \xleftarrow{\$} \Sigma_1.\mathcal{M}$

$C_1 = Enc_1(K_1, K_2)$

$C_2 = Enc_2(K_2, M)$

ret (C_1, C_2)

$\Sigma^*.Enc : \mathcal{K}^* \times \mathcal{M}^* \to \mathcal{C}^*$

$(K_1, M_2) \to (C_1, C_2) \in (\mathcal{C}_1 \times \mathcal{C}_2)$

$\hookrightarrow \in \mathcal{K}_1$ $\hookrightarrow \in \mathcal{M}_2$

↳ need $\Sigma_1.\mathcal{M} = \Sigma_2.\mathcal{K}$

- use, e.g., Σ_1 slow, Σ_2 fast, so use Σ_2 for large msg M

Thm. If Σ_1 & Σ_2 are OTS, then Σ^* is OTS.

2B - 1/15

SKE Recap

+ syntax: $\mathcal{K}, \mathcal{M}, \mathcal{C}$
  - $Enc(K, M) \mapsto C$ (rand)
  - $Dec(K, C) \mapsto M$ (deterministic)
+ correctness: $\forall M \in \mathcal{M}, K \in \mathcal{K}, P[Dec(K, Enc(K, M)) = M] = 1$
+ One-Time Security
  - SKE Σ has OTS iff $\mathcal{L}^{\Sigma}_{OTS\text{-}real} \equiv \mathcal{L}^{\Sigma}_{OTS\text{-}rand}$

Hybrid Encryption

- start w/ SKE schemes Σ_1, Σ_2.
  - $Enc_2 : \mathcal{K}_2 \times \mathcal{M}_2 \to \mathcal{C}_2$
    - want to encrypt message here
    - use Σ_1 to encrypt Σ_2's key
  - $Enc_2 : \mathcal{K}_1 \times \mathcal{M}_1 (\supseteq \mathcal{K}_2) \to \mathcal{C}_1$; $\mathcal{K}_2 \subseteq \mathcal{M}_1$
- now we create Σ^*: $\mathcal{K}^* = \mathcal{K}_1$, $\mathcal{M}^* = \mathcal{M}_2$, $\mathcal{C}^* = \mathcal{C}_1 \times \mathcal{C}_2$

$Enc(K^*, M^*)$:

$K_2 \xleftarrow{\$} \mathcal{K}_2$

$C_1 \leftarrow Enc_1(K^*, K_2)$

$C_2 \leftarrow Enc_2(K_2, M)$

ret (C_1, C_2)

Thm. If Σ_1 & Σ_2 have OTS, then Σ^* has OTS.

Pf.

$\mathcal{L}^{\Sigma^*}_{\text{OTS-real}}$

$\underline{Enc^*(m)}$:
$K_1 \xleftarrow{\$} \mathcal{K}_1$
$K_2 \xleftarrow{\$} \mathcal{K}_2$
$C_1 = Enc_1(K_1, K_2)$
$C_2 = Enc_2(K_2, m)$
ret (C_1, C_2)

$\equiv$ Transform to:

$\underline{Enc^*(m)}$:
$K_2 \xleftarrow{\$} \mathcal{K}_2$
$C_1 = OTS.Enc_1(K_2)$
$C_2 = \Sigma_2.Enc_2(K_2, m)$
ret (C_1, C_2)

$\diamond$

$\mathcal{L}^{\Sigma_1}_{\text{OTS-real}}$

$\underline{Enc_1(m)}$:
$K_1 \xleftarrow{\$} \mathcal{K}_1$
$C = Enc(K_1, m)$
ret C

- but we can swap $\mathcal{L}^{\Sigma_1}_{\text{OTS-real}}$ for $\mathcal{L}^{\Sigma_1}_{\text{OTS-rand}}$ b/c it is OTS!
  - ↳ fold this back in:

$\mathcal{L}^{\Sigma^*}_{\text{OTS-real}} \equiv$

$\underline{Enc^*(m)}$:
$K_2 \xleftarrow{\$} \mathcal{K}_2$
$C_1 \xleftarrow{\$} \mathcal{C}_1$
$C_2 = Enc_2(K_2, m)$
ret (C_1, C_2)

do the same for Enc_2

$\equiv$

$\underline{Enc^*(m)}$:
$C_1 \xleftarrow{\$} \mathcal{C}_1$
$C_2 \xleftarrow{\$} \mathcal{C}_2$
ret (C_1, C_2)

$\equiv \mathcal{L}^{\Sigma^*}_{\text{OTS-rand}}$ ▨

New Schemes

+ problems w/ OTP?
  - ↳ $|K| = |m|$
  - ↳ one-time only

↳ malleability
↳ $Dec(K, C \oplus \Delta) = M \oplus \Delta$ → predictable change = bad

- any OTS scheme must have $|K| \geq |M|$
  ↳ long keys are inherent
- the one-time only part is also inherent
  ↳ e.g. Eve sees $(M_1, C_1), \ldots, (M_{1000}, C_{1000})$, wants to dec C^*
    ↳ inefficient adv. can succeed w/ good pr.
    ↳ efficient has low but nonzero pr.

- so we must relax security
  ↳ any attack must have huge cost or tiny pr. success
  ↳ "computational security"

+ concrete approach: pick big/small #
  ↳ e.g., 2^{128}, 2^{-128}
  ↳ or take ratio of time & pr: $\frac{T}{p} \geq 2^{256}$

+ asymptotic approach (complexity theory)
Def. $f, g: \mathbb{N} \to \mathbb{R}$. $f(n)$ is $O(g(n))$ if $\exists c$ s.t. $f(n) \leq c \cdot g(n)$ excluding finite n.

- recall alg. A runs in $T_A(n)$ time (worst case) for inputs $|x| = n$.
Def. poly time: $T_A = O(p(n))$, p polynomial.

↳ so "huge" will mean not polynomial time

Def: negligible functions. a fn. $f: \mathbb{N} \to \mathbb{R}_+$ is negligible if $\forall$ poly p, $p(n) = O(1/f(n))$

- properties:
  - ↳ poly +/- poly = poly
  - ↳ negl + negl = negl
  - ↳ poly · negl = negl

3A - 1/20

Computational Security

- assume poly-time attack algos → e.g. $\frac{1}{2^{n/2}}$
  - ↳ negligible attack success ($negl(n) \to 0$ faster than $\frac{1}{poly(n)}$)

- what are these fn.s poly/negl. in?
  - ↳ the security parameter $\lambda \in \mathbb{Z} \geq 1$
    - ↳ often $\lambda = |K|$, or the length of the secret
    - ↳ we'll assume all algos just know λ

Def. $\mathcal{L}_1$ and $\mathcal{L}_2$ are computationally indistinguishable (denoted $\mathcal{L}_1 \approx \mathcal{L}_2$) if $\forall$ poly-time algos $\mathcal{A}$,

$$|P[\mathcal{A} \diamond \mathcal{L}_1 \Rightarrow \text{True}] - P[\mathcal{A} \diamond \mathcal{L}_2 \Rightarrow \text{True}]|$$

is negligible.

"distinguishing advantage" → the λ is implicit!

Lemma. ① $\mathcal{L}_1 \equiv \mathcal{L}_2 \Rightarrow \mathcal{L}_1 \approx \mathcal{L}_2$

② $\mathcal{L}_1 \approx \mathcal{L}_2 \Rightarrow \mathcal{L}_3 \diamond \mathcal{L}_1 \approx \mathcal{L}_3 \diamond \mathcal{L}_2$

③ $\mathcal{L}_1 \approx \mathcal{L}_2 \approx \mathcal{L}_3 \Rightarrow \mathcal{L}_1 \approx \mathcal{L}_3$

Pf (of 3).

Fix poly-time $\mathcal{A}$. Let $p_i = P[\mathcal{A} \diamond \mathcal{L}_i \Rightarrow T]$.

$|p_1 - p_2|$ and $|p_2 - p_3|$ are negl.

By the triangle inequality, $|p_1 - p_3| \leq |p_1 - p_2| + |p_2 - p_3|$ is negligible.

Claim. For hybrid arguments $\mathcal{L}_1 \approx \cdots \approx \mathcal{L}_{t(\lambda)}$, if $t(\lambda)$ is poly-time then $\mathcal{L}_1 \approx \mathcal{L}_{t(\lambda)}$.

Using Shorter Keys

- our short key K will be "stretched" by G
  - ↳ $Enc(K, m)$:
    - $y = G(K)$
    - ret $y \oplus m$
  - ↳ need $|y| > |K|$, G poly-time, possible decryption ($\Rightarrow G$ deterministic)
    - ↳ and must "look random"

Def: Pseudorandom Generators (PRG). $G: \{0,1\}^\lambda \to \{0,1\}^{\lambda + \ell}$ $\forall \ell \geq 1$, poly-time computable & deterministic is a secure PRG if $\mathcal{L}_{PRG\text{-}real} \approx \mathcal{L}_{PRG\text{-}rand}$.

$\mathcal{L}_{PRG\text{-}real}$:

PRG.Sample():
- $s \xleftarrow{\$} \{0,1\}^\lambda$ ← seed, assumed not known by $\mathcal{A}$
- ret $G(s)$

$\mathcal{L}_{PRG\text{-}rand}$:

PRG.Sample():
- $y \xleftarrow{\$} \{0,1\}^{\lambda + \ell}$
- ret y

- these are kinda dope
  - ↳ say $\ell = 100$. Then $\{0,1\}^{\lambda+100}$ is 2^{100} x larger than $2^{\lambda+100}$!
  - ↳ $P[y \xleftarrow{\$} \{0,1\}^{\lambda+100} \in G(s)\ \forall\ s] \leq 2^{-100}$!
  - ↳ so most of the codomain is untouched by G!
  - ↳ yet to efficient algos they look the same

E.g.

$G(s) = s \| 00000000$

Say $\mathcal{A}$ just checks if $z = PRG.Sample()$ ends in 00000

- ↳ $P[\mathcal{A} \diamond \mathcal{L}_{PRG\text{-}real} \Rightarrow T] = 1$
- ↳ $P[\mathcal{A} \diamond \mathcal{L}_{PRG\text{-}rand} \Rightarrow T] = 1/2^{\sigma}$ $\quad |1 - 1/2^{\sigma}|$ not negl!

- the psedo-OTP uses an extended $y = G(k)$ for $y \oplus m$.
  - ↳ how to prove security?

Def. SKE Σ has computational OTS if $\mathcal{L}^{\Sigma}_{OTS\text{-}real} \approx \mathcal{L}^{\Sigma}_{OTS\text{-}rand}$.

Thm. If G is a secure PRG, then psedo-OTP has COTS.

Pf.

OTS.Enc(m)	≡	OTS.Enc(m):	◇	PRG.Sample()
$k \xleftarrow{\$} \{0,1\}^{\lambda}$		$y = PRG.Sample()$		$s \xleftarrow{\$} \{0,1\}^{\lambda}$
$y = G(k)$		ret $y \oplus m$		ret $G(s)$
ret $y \oplus m$				

$\approx$

OTS.Enc(m):
$y = PRG.Sample()$
ret $y \oplus m$

$\diamond$

PRG.Sample()
$y \xleftarrow{\$} \{0,1\}^{\lambda+\ell}$
ret y

$\equiv$

OTS.Enc():
$y \xleftarrow{\$} \{0,1\}^{\lambda+\ell}$
ret $y \oplus m$

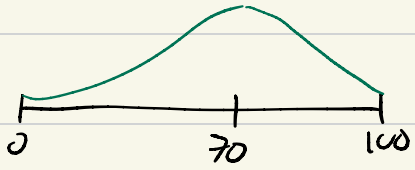
$=$

OTS.Enc(m):
$C \leftarrow \{0,1\}^{\lambda+\ell}$
ret C

$\equiv \mathcal{L}_{OTP\text{-}rand}$

3B - 1/22

Prof's exam distribution:



PRGs

- we previously showed that if G is a secure PRG, then pseudo-OTP has OTS.
- we'll later see that PRFunctions give many-time security
  - ↳ and that PRG $\Leftrightarrow$ PRF
  - ↳ crypto is a/b identifying important abstractions like PRGs that are the cruxes of the problems

- recall that $G(s) \approx y \xleftarrow{\$} \{0,1\}^{\lambda+\ell}$ for $\mathcal{L}^G_{\text{PRG-real}}$ to be a PRG

Claim. Suppose $G: \{0,1\}^\lambda \to \{0,1\}^{\lambda+\ell}$ is a secure PRG. Say $H(s) = G(\text{rev}(s))$. Then H is a secure PRG

Pf.

$\text{Samp}_H()$:	$\equiv$	$\text{Samp}_H()$:	$\equiv$	$\text{Samp}_H()$:	$\diamond$	$\text{Samp}_G()$:	$\approx$	$\text{Samp}_G()$:
$s \xleftarrow{\$} \{0,1\}^\lambda$		$z \xleftarrow{\$} \{0,1\}^\lambda$		$y = \text{Samp}_G()$		$s \xleftarrow{\$} \{0,1\}$		$y \xleftarrow{\$} \{0,1\}^{\lambda+\ell}$
$z = \text{rev}(s)$		$y = G(z)$		ret y		ret $G(s)$		ret y
$y = G(z)$		ret y						
ret y								

b/c $\mathcal{L}^G_{\text{real}} \approx \mathcal{L}^G_{\text{rand}}$

- PRG existence implies $P = NP$

PRG E.g.s

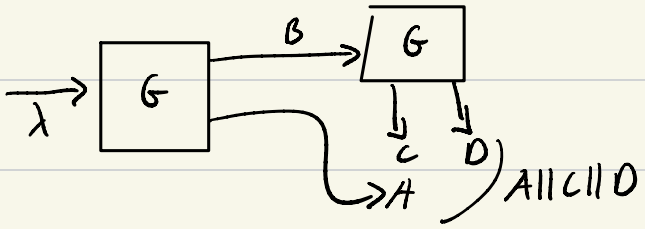
+ Subset Sum Generator
  - ↳ pick 512 random 1024-bit strings $w_1, \ldots, w_{1024}$
  - ↳ $G_{w_1,\ldots,w_{1024}}(s_1, \ldots, s_{512}) = \sum_{i=1}^{512} s_i \cdot w_i \bmod 2^{1024}$
    - ↳ we think this is a PRG, but proving it solves P = NP

- what if we want PRG output length flexibility?

E.g.

Suppose $G: \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ ("length doubling")

Let's construct $H: \{0,1\}^\lambda \to \{0,1\}^{3\lambda}$



$H(s)$:
$A \| B = G(s)$
$C \| D = G(B)$
ret $A \| C \| D$

Pf.

$Samp_H()$:	≈	$Samp_H()$:	≡	$Samp_H()$:	≈	$Samp_H()$:	$= L^H_{PRG\text{-}rand}$
$s \xleftarrow{\$} \{0,1\}^\lambda$		$A \| B \xleftarrow{\$} \{0,1\}^{2\lambda}$		$A \xleftarrow{\$} \{0,1\}^\lambda$		$A \xleftarrow{\$} \{0,1\}^\lambda$	
$A \| B = G(s)$		$C \| D = G(B)$		$B \xleftarrow{\$} \{0,1\}^\lambda$			
$C \| D = G(B)$		ret $A \| C \| D$		$C \| D = G(A)$		$C \| D \xleftarrow{\$} \{0,1\}^{2\lambda}$	
ret $A \| C \| D$				ret $A \| C \| D$		ret $A \| C \| D$	

- if we instead had ret $A \| B \| C \| D$, we could break it by checking if $G(B) = C \| D$

- so far we've seen stuff like "say G is a secure PRG. Then..."
  1. H is always a PRG
  2. H is never a PRG
  3. H is sometimes a PRG
     - ↳ H is secure only for some G.
     - ↳ e.g. $H(s) = G(0^\lambda \| s)$ is sometimes

Midterm 2/9, 7-8:20 Kent 102?
120 : thought } W1-4 content, PSets 1-4

4A - 1/27

- tod: 6.1 - 6.3, Ch. 4 lemmas

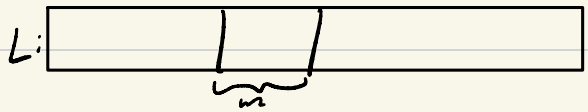
Pseudorandom Functions

↳ will let us use keys >1 time

- recall for PRGs, $|G(s)| = poly(|s|)$. → pset poly & extension blc efficiency
  ↳ what if we want more?
    ↳ exp. long string is unusable, but what if we just want the ith or jth element?

- $F: \{0,1\}^\lambda \times \{0,1\}^n \to \{0,1\}^m$
  key K, input X, output Y
  ↳ $n \approx \lambda, 2\lambda, \frac{\lambda}{2}$

$\mathcal{L}_{prf\text{-}rand\text{-}HUGE}$



for $X \in \{0,1\}^n$
  $L[X] \leftarrow\$ \{0,1\}^m$

Query(x):
  ret L[x]

Def. F is a secure PRF if $\mathcal{L}^F_{prf\text{-}real} \approx \mathcal{L}^F_{prf\text{-}rand}$

$\mathcal{L}_{prf\text{-}real}$
$K \leftarrow\$ \{0,1\}^\lambda$
Query(x):
  ret F(K, x)

$\approx$

$\mathcal{L}_{prf\text{-}rand}$
L: init empty
Query(x):
  if L[x] null:
    $L[x] \leftarrow\$ \{0,1\}^m$
  ret L[x]

E.g.

Let G be a secure PRG. Define $F(K, X) = G(K) \oplus X$

↳ basically pseudo-OTP

F is not a PRF.

$\mathcal{A}$
```
Y = Query(0^n)
Y' = Query(Y)
if Y' = 0^n:
  ret T
ret F
```

$$P[\mathcal{A} \diamond \mathcal{L}_{prg\text{-}real} \Rightarrow T] = P[G(K) \oplus G(K) = 0^n] = 1$$

$$P[\mathcal{A} \diamond \mathcal{L}_{prg\text{-}rand} \Rightarrow T] = P[Query(0^n) = 0^n] + P[Query(Y) = 0^n] \leq 2 \cdot 2^{-n}$$

E.g.

$F(K, X) = G(X) \oplus K$ also doesn't work

e.g. if $\tilde{G} = \begin{cases} 0^n & \text{if } s = 0^n \\ G(s) & \text{o.w.} \end{cases}$, this is valid PRG but breaks F.

- PRG $\Rightarrow$ PRF proof in book

- let's construct PRF

Claim. Let PRF $F: \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^\lambda$. Let $H(K_1 \| K_2, X)$: $Y = F(K_1, X)$; ret $F(K_2, Y)$. Then H is a PRF.

Pf.

introduce dict. ✓ from $\mathcal{L}_{prf\text{-}rand}$

$\mathcal{L}_{prf\text{-}real}$
```
K1 <-$ {0,1}^λ
K2 <-$ {0,1}^λ
Query(X):
  Y = F(K1, X)
  ret F(K2, Y)
```
$\equiv$
```
K1 <-$ {0,1}^λ
K2 <-$ {0,1}^λ
Query(X):
  if L_H[X] null:
    Y = F(K1, X)
    L[X] = F(K2, Y)
  ret L[X]
```
$\approx$
```
K2 <- $
Query(X):
  if L_H[X] null:
    Y = Query_F(X)
    L[X] = F(K2, Y)
  ret L[X]
```

$\diamond \mathcal{L}^F_{prf\text{-}real}$

$\diamond \mathcal{L}^F_{prf\text{-}real}$

$K_2 \leftarrow \$$

$\underline{\text{Query}_H(X)}$:

if $L[X]$ null:

$\approx$ $Y \xleftarrow{\$} \{0,1\}^\lambda$

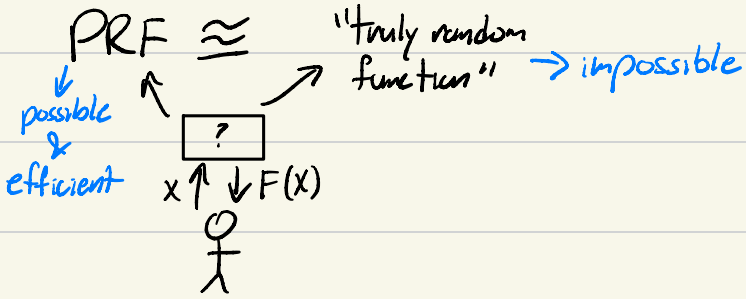
$L[X] = F(K_2, Y)$

ret $L[X]$

4B - 1/29

Intuition

- our definitions basically follow real world $\approx$ random world
  - ↳ means any alg. can't tell the real & random apart
- e.g. for COTS,
  - ↳ real = encrypt a msg
  - ↳ rand = random junk
- PRG:
  - ↳ real: sample seed & stretch it
  - ↳ rand: uniform @ random
- PRF diagram:



$\mathcal{L}_{prf\text{-}rand}$

```
Query(X):
if L[X] null:
  L[X] ←$ {0,1}^m
ret L[X]
```

$\approx$

$\mathcal{L}_{prf\text{-}real}$

```
K ←$ {0,1}^λ
Query(X):
  ret F(K, X)
```

→ this is basically setting "which" function we choose

Tip: to break a constructed PRF that uses PRF F, try to cause repeated inputs to F.

New PRF

Claim. F is a PRF w/ λ-bit inputs/outputs. Define $H(K_1 \| K_2, X)$:

```
Y = F(K1, X)
ret F(K2, Y)
```

If F is a secure PRF, then H is as well.

without replacement:
$R \xleftarrow{\$} \{0,1\}^\lambda \setminus S$
$S = S \cup \{R\}$

Pf.

$\mathcal{L}^H_{\text{prf-real}}$

$K_1 \| K_2 \xleftarrow{\$} \{0,1\}^{2\lambda}$
Query(X):
$Y = F(K_1, X)$
ret $F(K_2, Y)$

$\approx$

Query$_H$(X):
if $L_1[X]$ null:
$L_1[X] \xleftarrow{\$} \{0,1\}^\lambda$
if $L_2[Y]$ null:
$L_2[Y] \xleftarrow{\$} \{0,1\}^\lambda$
return $L_2[Y]$

$\approx$ ↓ requires later proof

Query$_H$(X):
if $L_1[X]$ null:
$L_1[X] \xleftarrow{\$} \{0,1\}^\lambda$ w/o replacement
if $L_2[Y]$ null:
$L_2[Y] \xleftarrow{\$} \{0,1\}^\lambda$
return $L_2[Y]$

all these Y's should be $L_1[X]$'s

$\equiv$

Query$_H$(X):
if $L_1[X]$ null:
$L_1[X] \xleftarrow{\$} \{0,1\}^\lambda$ w/o repl.
$L_2[Y] \xleftarrow{\$} \{0,1\}^\lambda$
return $L_2[Y]$

$\equiv$

Query$_H$(X):
if $L_H[X]$ null:
~~if $L_1[X]$ null:~~ does same thing
~~$L_1[X] \xleftarrow{\$} \{0,1\}^\lambda$ w/o repl.~~ → doesn't matter now
$L_2[Y] \xleftarrow{\$} \{0,1\}^\lambda$
$L_H[X] = L_2[L_1[X]]$
ret $L_H[X]$

b/c each $L_1[X]$ is only sampled once, $L_2[Y] = L_2[L_1[X]]$ will only be null if $L_1[X]$ is

$\equiv$

Query$_H$(X):
if $L_H[X]$ null:
$L_H[X] \xleftarrow{\$} \{0,1\}^\lambda$
ret $L_H[X]$

$\equiv \mathcal{L}^H_{\text{prf-rand}}$

Replacement

Sample q items $X_1, \ldots, X_q$ uniformly from a set of size N.
$P[\exists i \neq j \text{ s.t. } X_i = X_j] \leq \frac{q^2}{N}$. If q poly & N exp., $\frac{q^2}{N}$ negligible!

Lemma.

Sample():
$R \leftarrow \{0,1\}^\lambda$
ret R

$\approx$

Sample():
$R \xleftarrow{\$} \{0,1\}^\lambda \setminus S$
$S = S \cup \{R\}$
ret R

Lemma: bad event technique. If ① L_1, L_2 bool BAD, ② L_1, L_2 identical unless BAD true, and ③ once BAD is true it remains true, then $|P[\mathcal{A} \diamond L_1 \Rightarrow T] - P[\mathcal{A} \diamond L_2 \Rightarrow T]| \leq P[\mathcal{A} \diamond L_1 \text{ triggers BAD}]$

P_1 P_2

Replacement Proof.

Sample(): $R \xleftarrow{\$} \{0,1\}^\lambda$ ret R	$\equiv$	Sample $R \xleftarrow{\$} \{0,1\}^\lambda$ if $R \in S$ BAD = True $S = S \cup \{R\}$ ret R	$\approx$	Sample $R \xleftarrow{\$} \{0,1\}^\lambda$ if $R \in S$ BAD = True $R \xleftarrow{\$} \{0,1\}^\lambda \setminus S$ $S = S \cup \{R\}$ ret R	$\equiv$	Sample $R \xleftarrow{\$} \{0,1\}^\lambda \setminus S$ $S = S \cup \{R\}$ ret R

By the birthday thing, $P[BAD]$ is negligible, so no efficient $\mathcal{A}$ exists. ∎

Bad Event Proof.

$$P_i[T] = P_i[T|B]\,P_i[B] + \underbrace{P_i[T|\bar{B}]\,P_i[\bar{B}]}_{\text{same in } L_1 \& L_2}$$

$$\Rightarrow |P_1[T] - P_2[T]| = |P_1[T|B]\,P_1[B] - P_2[T|B]\,P_2[B]|$$

$$= P[B] \cdot \underbrace{|P_1[T|B] - P_2[T|B]|}_{\leq 1}$$

$$\leq P[B]$$

Midterm:
- break proposed PRG
- T/F which schemes are OTS/COTS/neither
- prove simple claim
- prove small part of big claim
- if we change this to that, where does the proof break?

5A - 2/3

Recap

- OTS:

OTS.Enc(m):
$K \xleftarrow{\$} \mathcal{K}$
$C = Enc(K, m)$
ret C

$\approx$

OTS.Enc(m):
$C \xleftarrow{\$} \mathcal{C}$
ret C

→ "one-time" secrecy b/c key chosen every time
↳ but we want to be able to reuse keys!

First Attempt

$K \xleftarrow{\$} \mathcal{K}$
Enc(m):
  $C = Enc(K, m)$
  ret C

- adv. chooses the M b/c we don't want to make any assumptions a/b it
  ↳ "chosen plaintext attack"

WRONG!

Def: ~~CPA-Security~~. Let Σ be a SKE encryption scheme $(\mathcal{K}, \mathcal{C}, \mathcal{M}, Enc, Dec)$. Σ has CPA security iff

$$\mathcal{L}^{\Sigma}_{\text{CPA-real}} \triangleq \boxed{\begin{array}{l} K \xleftarrow{\$} \mathcal{K} \\ \text{CPA.Enc(m)} \\ \hline C = Enc(K, m) \\ \text{ret C} \end{array}} \approx \boxed{\begin{array}{l} \text{CPA.Enc(m)}: \\ C \xleftarrow{\$} \mathcal{C} \\ \text{ret C} \end{array}} \triangleq \mathcal{L}^{\Sigma}_{\text{CPA-rand}}$$

- what are the problems w/ the above?

① Enc can't be deterministic

↳ say $\mathcal{A}$ encrypts the same M twice; $\mathcal{A} \diamond \mathcal{L}_{\text{CPA-real}}$ always returns True, $P[\mathcal{A} \diamond \mathcal{L}_{\text{CPA-rand}} \Rightarrow T] = \frac{1}{|\mathcal{C}|}$

↳ need $|\mathcal{C}| \geq 2$ for 2 M's, otherwise scheme not correct

↳ thus, Enc must be randomized

② Let $M = \{\text{all files up to 1 TB}\}$, $M_1 = 100\text{ B}$, $M_2 = 50\text{ GB}$.

↳ but $C_1, C_2 \xleftarrow{\$} \mathcal{C}$, and $E[|C_1|] = E[|C_2|] \Rightarrow E[|Enc(K, M_1)|] = E[|Enc(K, M_2)|]$

↳ so we have to blow up $|C_1|$?

↳ and if $\mathcal{C} = \{0,1\}^*$, "can't uniformly sample an integer"

↳ so we leak $|M|$

Def: CPA-Security. Let Σ be a SKE encryption scheme $(\mathcal{K}, \mathcal{C}, M, Enc, Dec)$. Σ has CPA security iff

$$\mathcal{L}^{\Sigma}_{\text{CPA-real}} \triangleq \boxed{\begin{array}{l} K \xleftarrow{\$} \mathcal{K} \\ \text{CPA.Enc}(m) \\ \hline C = Enc(K, m) \\ \text{ret } C \end{array}} \approx\approx \boxed{\begin{array}{l} \text{CPA.Enc}(m): \\ \hline C \xleftarrow{\$} \mathcal{C}(|m|) \\ \text{ret } C \end{array}} \triangleq \mathcal{L}^{\Sigma}_{\text{CPA-rand}}$$

where $\mathcal{C}(\ell)$ is all the ctxts for length ℓ messages, i.e.,

$\mathcal{C}(\ell) = \{C : \exists K, M \text{ s.t. } |M| = \ell, P[Enc(K, M) = C] > 0\}$

↳ this point doesn't matter for fixed-length M's b/c the length is implicitly leaked

E.g. Proving not CPA

$K \xleftarrow{\$} \{0,1\}^\lambda$
$K' \xleftarrow{\$} \{0,1\}^\lambda$
Enc'(K, M):
  $C = K \oplus K'$
  $C' = K' \oplus M$
  ret (C, C')

$\mathcal{A}$ | $M = 0^\lambda$
$C_1 \| C_1' = CPA.Enc(M)$
$C_2 \| C_2' = CPA.Enc(M)$
if $C_1 \oplus C_1' = C_2 \oplus C_2'$:
  ret T
ret F

$\rightarrow P[\mathcal{A} \circ L_{CPA\text{-}real} \Rightarrow T] = 1$

$P[\mathcal{A} \circ L_{CPA\text{-}rand} \Rightarrow T] = 2^{-\lambda}$

Def: PRF-OTP

Let F be a PRF w/ λ-bit inputs & n-bit outputs.

Enc(K, M):
  $R \xleftarrow{\$} \{0,1\}^\lambda$
  $S = F(K, R) \oplus M$
  ret $R \| S$

Dec(K, R||S):
  $M = F(K, R) \oplus S$
  ret M

- $\mathcal{K} = \{0,1\}^\lambda$ (prf keys)
- $\mathcal{M} = \{0,1\}^n$
- $\mathcal{C} = \{0,1\}^{\lambda+n}$

Thm.

If F is a secure PRF, the PRF-OTP has CPA-security.

Pf.

$K \xleftarrow{\$} \mathcal{K}$
CPA.Enc(X):
  $R \xleftarrow{\$} \{0,1\}^\lambda$
  $Y = F(K, R)$
  $S = Y \oplus M$
  ret $R \| S$

PRF sec. $\approx$

CPA.Enc(X):
  $R \xleftarrow{\$} \{0,1\}^\lambda$
  if L[R] null:
    $L[R] \xleftarrow{\$} \{0,1\}^n$
  $Y = L[R]$
  $S = Y \oplus M$
  ret $R \| S$

$\approx$ w/ & w/o repl. lemma $\approx$

CPA.Enc(X):
  $R \xleftarrow{\$} \{0,1\}^\lambda$ w/o repl.
  if L[R] null:
    $L[R] \xleftarrow{\$} \{0,1\}^n$
  $Y = L[R]$
  $S = Y \oplus M$
  ret $R \| S$

$\approx$

CPA.Enc(X):
  $R \xleftarrow{\$} \{0,1\}^\lambda$ w/o repl.
  $Y \xleftarrow{\$} \{0,1\}^n$
  $S = Y \oplus M$
  ret $R \| S$

OTP! perfect secrecy

by perfect secrecy of OTP

$\equiv$

CPA.Enc(x):
$R \xleftarrow{\$} \{0,1\}^\lambda$ w/o repl.
$S \leftarrow \{0,1\}^n$
ret $R \| S$

$\approx$

CPA.Enc(x):
$R \xleftarrow{\$} \{0,1\}^\lambda$
$S \leftarrow \{0,1\}^n$
ret $R \| S$

w/ & w/o replacement lemma

$\equiv \mathcal{L}_{\text{CPA-rand}}$

5B- 2/5

CPA

- recall our definition of CPA-security:

$\mathcal{L}^{\Sigma}_{\text{CPA-real}}$

$K \xleftarrow{\$} \mathcal{K}$

CPA.Enc(M):

$C = Enc(K, M)$

ret C

$\approx$

$\mathcal{L}^{\Sigma}_{\text{CPA-rand}}$

CPA.Enc(M):

$C \xleftarrow{\$} \mathcal{C}(|M|)$ → $M = \{0,1\}^*$

ret C

- we want to perform **domain extension**: taking our scheme from $\{0,1\}^n$ to $\{0,1\}^*$

Problem: Given Σ with $\mathcal{M} = \{0,1\}^n$, construct Σ^* w/ $\mathcal{M}^* = (\{0,1\}^n)^*$

↳ $M^* = M_1 || M_2 || \dots || M_\ell$: ℓ n-bit blocks

- Soln: $Enc^*(K, M_1 || \dots || M_\ell) = Enc(K, M_1) || \dots || Enc(K, M_\ell)$

Claim. If Σ CPA-secure, Σ^* is CPA-secure.

Pf. $\mathcal{L}^*_{\text{CPA-real}}$

$K \xleftarrow{\$} \mathcal{K}$

CPA.Enc(M):

$C = Enc^*(K, M)$

ret C

$\equiv$

CPA.Enc(M) ($M_1 || \dots || M_\ell$)

for $i = 1 \dots \ell$

$C_i = $ CPA.Enc(M_i)

ret $C_1 || \dots || C_\ell$

$\diamond \mathcal{L}_{\text{CPA-real}} \approx$

CPA.Enc(M)

for $i = 1 \dots \ell$

$C_i = $ CPA.Enc(M_i)

ret $C_1 || \dots || C_\ell$

$\diamond \mathcal{L}_{\text{CPA-rand}}$

$\equiv$

CPA.Enc(M):

for $i = 1, \dots, \ell$

$C_i \xleftarrow{\$} \{0,1\}^{n+\lambda}$

ret $C_1 || \dots || C_\ell$

$\equiv$

CPA.Enc(M):

$C \xleftarrow{\$} \{0,1\}^{\ell(n+\lambda)}$

ret C ($\in \mathcal{C}(\ell)$ or $\mathcal{C}(n\ell)$ depending on the defn)

- Let F be a PRF w/ input & output length n, $M = \{0,1\}^*$
  ↳ we prev. showed $G(R) \triangleq F(K,R) \| F(K,R+1) \| \dots \| F(K,R+\ell)$ is secure PRG

CTR Enc(K, M):
$R \xleftarrow{\$} \{0,1\}^n$
PAD = use CTR PRG to get $|M|$ pseudorandom bits w/ seed R
ret $R \| (PAD \oplus M)$

↳ CTR construction PRG

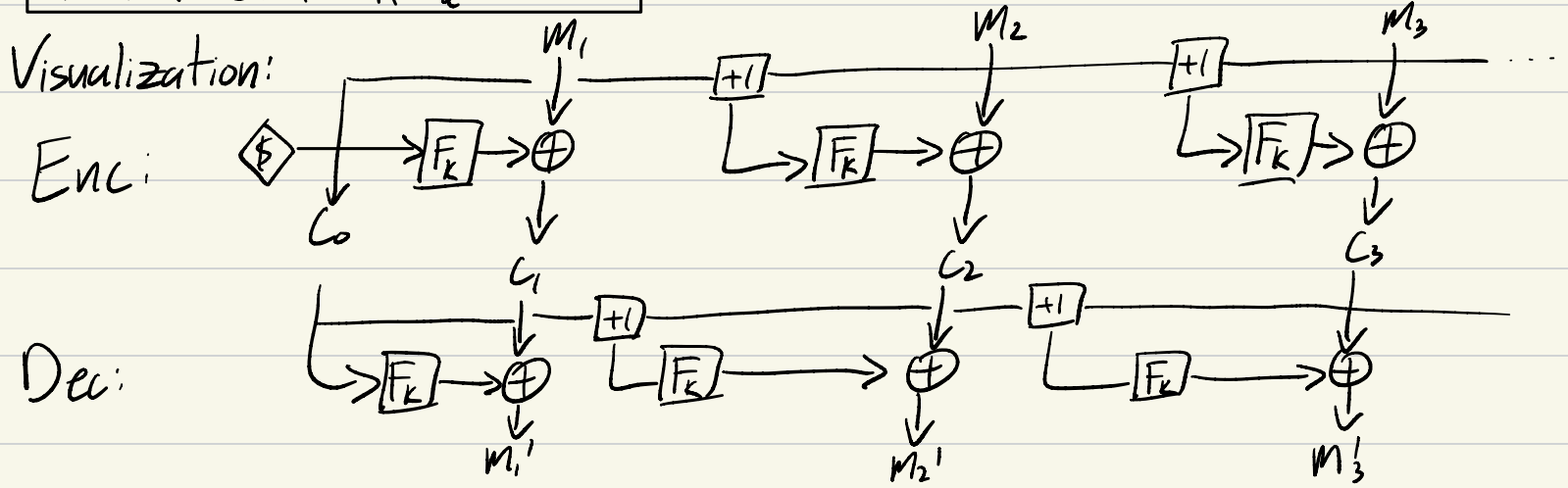
→ won't prove security bc uninteresting

- we'll use this to motivate **CTR-mode encryption**
  ↳ Let $M = (M_1 \| \dots \| M_\ell)$, w/ M_ℓ possibly $< n$ bits

Enc(K, M):
$C_0 \xleftarrow{\$} \{0,1\}^n$
for $i = 1, \dots, \ell$:
  $C_i = F(K, C_0 + i - 1) \oplus M_i$
return $C_0 \| \dots \| C_\ell$

→ equivalent to the above Enc, just w/o making reference to the PRG G

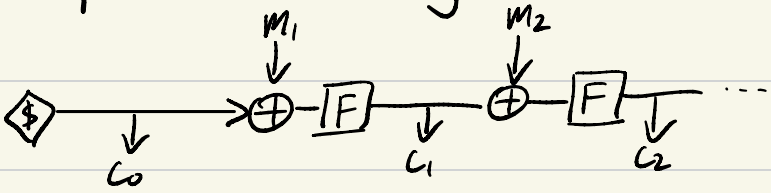
Visualization:



Topics we skipped

+ block ciphers
  ↳ just pseudorandom permutations, i.e. invertible PRF
    ↳ i.e. $F(K, X) \to Y$, $F^{-1}(K, Y) \to X$; F must be $\{0,1\}^n \to \{0,1\}^n$

- AES is most popular block cipher
- using block cipher for enc. scheme = enc. mode
  ↳ e.g. cipher block chaining (CBC):



Editing Eve

- $OTP(K, m) = K \oplus M$
- say Eve does $C \oplus (00 \ldots 01) = C'$
  ↳ then the last bit of decrypted C' will be flipped

Def: Malleability. changing the ctxt leads to a predictable change in the plaintext.

- what if an adversary edited the CTR-mode encryption?
  ↳ $(c_0 || c_1 || c_2 || c_3) \to (c_0 || c_1 \oplus \Delta || c_2 || c_3)$
  ↳ $\Delta \in \{0,1\}^n$
  ↳ $Dec(K, c_1 \oplus \Delta_1 || \ldots || c_\ell \oplus \Delta_\ell) = m_1 \oplus \Delta_1 || \ldots || m_\ell \oplus \Delta_\ell$ ← simple & predictable

- malleability opens up attacks (claim 9.1.1)

- say this null oracle exists:

```
K ←$ K
NullOracle(C):
  M = Dec(K, C)
  ret 0x00 in M
```

+ the attack:
  - take first byte of M (in the ctxt), XOR w/ all possible bytes
    ↳ the byte that makes the null oracle return true is the byte of the msg!
  - do for all bytes ($256 \cdot |m|$)

zoom lectures next week

6A - 2/10

- recall

$\mathcal{L}_{\text{CPA-real}}$
$K \xleftarrow{\$} \mathcal{K}$
CPA.Enc(m):
ret Enc(K, m)

→ sort of strange the attacker can chose m

- also discussed malleability
  - ↳ allowed for attack where adv. sends many $\hat{C}_1, \hat{C}_2, \ldots$ and observes Bob's response
    - ↳ if $\hat{C}_j$'s response does something strange, adv. can learn something a/b the message (null oracle attack)
- adv. has more power, so security should be harder
  - ↳ we assume adv. sees some info a/b $Dec(K, \hat{C})$
  - ↳ this is basically worst case: Bob decrypts msg. & reveals it

Def: CCA Security. Σ has CCA security if

$\mathcal{L}_{\text{CCA-real}}$
$K \xleftarrow{\$} \Sigma.\mathcal{K}$
CCA.Enc(m):
ret Σ.Enc(m)
CCA.Dec(C)
ret Σ.Dec(m)

$\approx$

$\mathcal{L}_{\text{CCA-rand}}$
$K \xleftarrow{\$} \mathcal{K}$
CCA.Enc(m):
$C \leftarrow \mathcal{C}(\lvert m \rvert)$
$D[C] = m$
ret C
CCA.Dec(C):
if D[C] not null:
ret D[C]
ret Σ.Dec(K, C)

- Random world's returned ctxts = challenge ctxts
  ↳ idea: keep record of what messages the adv. sent correspond to which challenge ctxts

E.g.

PRF $F: \lambda \times \lambda \to \lambda$

$\mathcal{K} = b^\lambda$, $\mathcal{M} = b^\lambda$, $\mathcal{C} = b^{2\lambda}$

- We will break:

$\underline{Enc(K, m)}$:
$R \xleftarrow{\$} \{0,1\}^\lambda$
$S = F(K, R) \oplus m$
ret $R \| S$

$\underline{Dec(K, R \| S)}$:
ret $F(K, R) \oplus S$

First, note $Dec(K, (R \| S) \oplus \Delta) = Dec(K, R \| S) \oplus \Delta$

$\underline{\mathcal{A}}$
$m = 0^\lambda$, $\Delta = 1^\lambda$
$R \| S = CCA.Enc(m)$
$m' = CCA.Dec(R \| (S \oplus \Delta))$
ret $m' = m \oplus \Delta$

the game w/ CCA is to query Dec w/ a ctxt that wasn't the output of Enc

$P[\mathcal{A} \diamond \mathcal{L}_{CCA\text{-real}} \Rightarrow T] = 1$

$$P[\mathcal{A} \diamond \mathcal{L}_{\text{CCA-rand}} \Rightarrow T] = P[F(K,R) \oplus S = M \oplus \Delta]$$
$$= P[S = M \oplus \Delta \oplus F(K,R)] = 2^{-\lambda}$$

S unif rand, $\perp$ of $M, \Delta, F(K,R)$

Alt: $P[\mathcal{A} \diamond \mathcal{L}_{\text{CCA-rand}} \Rightarrow T] = P[M = F(K,R) \oplus \Delta \oplus S]$

if we chose $M \xleftarrow{\$} b^\lambda$,

read in book ← this pr. is again $2^{-\lambda}$

E.g.

Say F is now invertible.

$\underline{\text{Enc}(K, M)}$:
$R \xleftarrow{\$} \{0,1\}^\lambda$
$S = F(K,R) \oplus F(K,M)$
ret $R \| S$

$\underline{\text{Dec}(K, R \| S)}$:
ret $F^{-1}(K, F(K,R) \oplus S)$

Let's look at two encryptions of M:
$R_1 \| F(K,R_1) \oplus F(K,M)$
$R_2 \| F(K,R_2) \oplus F(K,M)$

XOR these to get $F(K,R_1) \oplus F(K,R_2)$

add R_1

We get $R_1 \| F(K,R_1) \oplus F(K,R_2)$, the ctxt from $\text{Enc}(R_2)$!
↳ ⇒ $\text{Dec}(K, \ldots)$ should equal R_2

Formulize

$\mathcal{A}$:

$M = 0^\lambda$

$R_1 || S_1 = CCA.Enc(M)$

$R_2 || S_2 = CCA.Enc(M)$

$M' = CCA.Dec(R_1 || S_1 \oplus S_2)$

ret $M' = R_2$

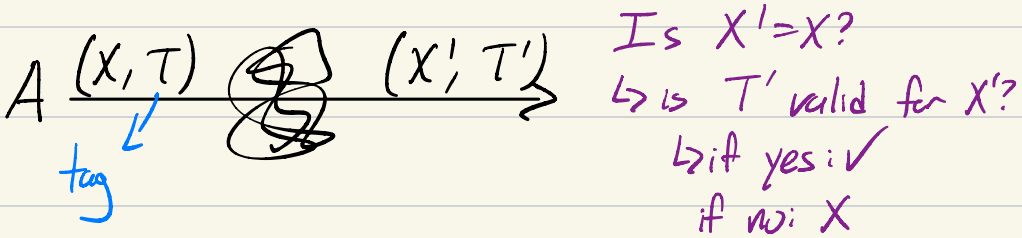
GB - 2/12

- recall the ctxts from Enc in CCA are "challenge ctxts"

Building CCA

- idea: if the adv. alr knows all the answers it can get from Dec, then Dec is useless to them & we basically have a CPA setting
  - ↳ goal: if adv. queries w/ non challenge ctxt, give error
    - ↳ authenticity

Def: MAC. message authentication code



- w/ a PRF: Alice sends $(X, F(k, X))$, Bob gets (X', T')
  - ↳ check: $T' = F(k, X')$?

Lemma. Suppose F is a secure PRF. Then:

$\mathcal{L}_{mac\text{-}real}$

$k \xleftarrow{\$} \{0,1\}^\lambda$

MAC.Reveal(X):
  ret F(k, X)

MAC.Guess(X, Y):
  ret Y = F(k, X)

$\approx$

$\mathcal{L}_{mac\text{-}ideal}$

MAC.Reveal(X):
  if L[X] null:
    $L[X] \xleftarrow{\$} \{0,1\}^\lambda$
  ret L[X]

MAC.Guess(X, Y):
  if L[X] null:
    ret False
  ret Y = L[X]

- In the ideal side, it is impossible to guess some X, Y before revealing X
  - ↳ i.e., $P[\mathcal{A} \diamond \mathcal{L}_{mac\text{-}real}^{(*)} \text{ guess}(X,Y) = T \text{ before reveal}(X)]$ is negl.
    - ↳ true by the above lemma, else we create B: run $\mathcal{A} \diamond \mathcal{L}$; if (*) ret T; else ret F
      which must be non-negl.

Encrypt-then-MAC

Let Σ be SKE scheme, F secure PRF w/ output length λ and input space containing $\Sigma.\mathcal{C}$

Enc$((K_e, K_m), M)$:

$C = \Sigma.\text{Enc}(K_e, M)$
$T = F(K_m, C)$
ret $C \| T$

Dec$((K_e, K_m), C\|T)$:

if $T \neq F(K_m, C)$:
  return $\perp$
ret $\Sigma.\text{Dec}(K_e, C)$

Spaces:

$$\mathcal{K} = \Sigma.\mathcal{K} \times \{0,1\}^\lambda, \quad \mathcal{M} = \Sigma.\mathcal{M}, \quad \mathcal{C}(\ell) = \Sigma.\mathcal{C}(\ell) \times \{0,1\}^\lambda$$

- why can't we just construct a C', pass it to MAC.Reveal(C'), then use that tag in an attack?
  - ↳ bc the CCA library has no MAC interface

Pf plan: ① remove K_E from CCA.Dec(C||T)
② use CPA security

Thm. Σ CPA, F PRF, then enc-then-mac is CCA secure

Pf.

$\mathcal{L}_{\text{CCA-real}}$

$K_E \xleftarrow{\$} \mathcal{K}$
$K_m \xleftarrow{\$} \{0,1\}^\lambda$
CCA.Enc(m):
  C = Enc(K_E, m)
  T = F(K_m, C)
  ret C||T
CCA.Dec(C||T):
  if T ≠ F(K_m, C):
    ret ⊥
  ret Dec(K_E, C)

≡

$\mathcal{L}_{\text{CCA-real}}$

$K_E \xleftarrow{\$} \mathcal{K}$
$K_m \xleftarrow{\$} \{0,1\}^\lambda$
CCA.Enc(m):
  C = Enc(K_E, m)
  T = F(K_m, C)
  D[C||T] = m
  ret C||T
CCA.Dec(C||T):
  if D[C||T] not null:
    ret D[C||T]
  if T ≠ F(K_m, C):
    ret ⊥
  ret Dec(K_E, C)

≡

$\mathcal{L}_{\text{CCA-real}}$

$K_E \xleftarrow{\$} \mathcal{K}$
CCA.Enc(m):
  C = Enc(K_E, m)
  T = MAC.Reveal(C)
  D[C||T] = m
  ret C||T
CCA.Dec(C||T):
  if D[C||T] not null:
    ret D[C||T]
  if not MAC.Guess(C, T)
    ret ⊥
  ~~ret Dec(K_E, C)~~

≈

◇ $\mathcal{L}_{\text{mac-real}}$

◇ $\mathcal{L}_{\text{mac-ideal}}$

Now notice that if D[C||T] is null, MAC.Reveal was never called, so MAC.Guess is False by construction. So we cross out the bottom line.

7A - 2/17

- we've been in symmetric/secret key setting
- now we'll deal w/ key dist.

Key Exchange

- Alice & Bob have no shared key
  ↳ adv. Eve listening to their communication

- basically we need cyclic groups

E.g.

arithmetic mod 13

$2 \xrightarrow{\times 2} 4 \xrightarrow{\times 2} 8, 3, 6, 12, 11, 9, 5, 10, 7, 1, 2$

↳ all #s mod 13 are here!

↳ $\{1, \dots, 12\}$ = group

↳ group operation = mult. mod 13

Def. Group is a set G w/ an operation $\cdot$ (multiplicative notation) satisfying:

* mult. not necessarily commutative; $a \cdot b \neq b \cdot a$

- closure: $\forall a, b \in G$, $a \cdot b \in G$
- identity: $\exists$ identity element $1 \in G$ s.t. $1 \cdot a = a = a \cdot 1$
- associativity: $(a \cdot b) \cdot c = a \cdot (b \cdot c)$
- inverses: $\forall a \in G$, $\exists a^{-1} \in G$ s.t. $a \cdot a^{-1} = 1$

- e.g. $5 \cdot 8 = 40 \cong 1 \mod 13$

E.g.

Is mult. mod 14 a group? $\{0, \dots, 13\}$

↳ no. $7 \cdot 2 \cong 0 \mod 14$

↳ also 7 has no inverse: $7x \cong_{14} 0, 7 \mod 14$

Def. $\mathbb{Z}_p^* = \{1, \dots, p-1\}$ is a group w/ mult. mod p $\forall$ primes p.

↳ usually $\mathbb{Z}_n$ denotes additive group (we don't do)

↳ no 0 bc 0 has no inverse

↳ generalized: $\mathbb{Z}_n^* = \{x : \gcd(x, n) = 1\} \subseteq [1, n-1]$

↳ size $|G|$ = order of group (e.g. $|\mathbb{Z}_p^*| \approx 2^\lambda$ for $p \approx 2^\lambda$)

Cyclic Group

Def. G is cyclic if $\exists\, g \in G$ s.t. $\forall\, a \in G$, $a = g^c$ (for some c). We say the element g generates the group.

↳ e.g. 2 generates $\mathbb{Z}_{13}^*$.

- e.g. 11 generates $\mathbb{Z}_{13}^*$

↳ $11 \cong_{13} -2$: $-2,\ 4,\ -8,\ 3,\ -6,\ 12,\ -11,\ 9,\ -5,\ 10,\ -7,\ 1$

(11, 5, 2, 8)

↳ e.g. 9 does not generate $\mathbb{Z}_{13}^*$: $9^k = (3^k)^2$; not all elements of $\mathbb{Z}_{13}^*$ are perfect squares

Prop. If g is a generator for G, then $\{g^0, g^1, \ldots, g^{m-1}\} = G$, where $m = |G|$.

↳ $g^{m-1} \cdot g = 1$; $g^{m-1} = g^{-1}$

$\Rightarrow g^a = g^{a \bmod |G|} = g^{a \bmod m}$

Prop. Given a generator g & element A, $\exists!\ a \in \{0, \ldots, m-1\}$ s.t. $g^a = A$.

Facts:

① $\forall$ prime p, $\mathbb{Z}_p^*$ cyclic group

② Finding generators of cyclic groups is efficient

↳ can check if $x^k = 1$ for $k < m-1 \Rightarrow x$ not generator in $\sim \log(m)$ time ($\log 2^\lambda = \text{poly}(\lambda)$)

↳ Lagrange's Thm: order of g = smallest k s.t. $g^k = 1$. Then, $\forall g \in G$, $\text{ord}(g)$ divides $|G|$

↳ only need to check that $g^{\frac{q}{t}} \not\equiv_p 1$ $\forall$ prime divisors q to show g generator

Discrete Log Problem

Given (g, A), find a s.t. $g^a = A$

↳ i.e., $\log_g A = \log_g(g^a) = a$

- This problem is thought to be Hard (for certain groups) ↗ $\mathbb{Z}_p^*$ hard

↳ best $\approx \sqrt{|G|} \approx 2^{\lambda/2}$ time

- luckily, exponentiating is fast
  ↳ given g, $a \in [0, n-1]$, find g^a
    ↳ $g^a = gg\cdots g$ a times $= 2^\lambda$ bad!

Repeated Squaring: write a in binary: $a = \sum_{i=1}^{\approx\lambda} b_i 2^i \Rightarrow g^a = g^{\sum_{i=1}^{\lambda} b_i 2^i}$

$= \prod_i g^{b_i 2^i} = \prod_{\substack{i \text{ s.t.} \\ b_i = 1}} g^{2^i}$

↳ only λ terms in this product, & each g^{2^λ} takes only λ squarings
  $\Rightarrow$ poly(λ) to exponentiate

- DLP is a conjectured **one-way function**:
  - computing f is easy (poly(λ))
  - computing f^{-1} is hard (exp(λ))
- OWFs are a core crypto primitive

Diffie-Hellman Key Exchange

- Alice & Bob sent msgs back & forth
  ↳ at the end, they return K_A & K_B, respectively
  ↳ want $K_A = K_B$, want security

Protocol. $\mathbb{G}$ cyclic group w/ generator $g \in \mathbb{G}$, order m.

① Alice & Bob sample $a \xleftarrow{\$} \mathbb{Z}_m$, $b \xleftarrow{\$} \mathbb{Z}_m$.

② Alice & Bob take $A = g^a$, $B = g^b$ resp., send to each other.

③ A: ret $B^a = (g^b)^a = g^{ab}$ $\Rightarrow K_A = K_b$

B: ret $A^b = (g^a)^b = g^{ab}$

Claim. If Alice & Bob execute honestly, given $A = g^a$ and $B = g^b$, Eve cannot distinguish $K = g^{ab}$ from $K \xleftarrow{\$} \mathbb{G}$.

$\mathcal{L}_{\text{ddh-real}}$
$a \xleftarrow{\$} \mathbb{Z}_m$
$b \xleftarrow{\$} \mathbb{Z}_m$
DDH.Get():
ret (g^a, g^b, g^{ab})

$\approx$

$\mathcal{L}_{\text{ddh-rand}}$
$a \xleftarrow{\$} \mathbb{Z}_m$
$b \xleftarrow{\$} \mathbb{Z}_m$
$c \xleftarrow{\$} \mathbb{Z}_m$
DDH.Get():
ret (g^a, g^b, g^c)

(DDH = decisional Diffie-Hellman assumption)

7B – 2/19

- $m = |\mathbb{G}|$ = order of group
- primes = p, q
- generator $g \in \mathbb{G}$ is s.t. $\{g^0, g^1, \ldots, g^{m-1}\} = \mathbb{G}$
  - ↳ $g^0 = 1$
  - ↳ $g^{m-1} \cdot g = g^m = g^0 = 1 \Rightarrow g^{m-1} = g^{-1}$
  - ↳ $g^a = g^{(a \bmod m)}$

Uniform Sampling: $a \xleftarrow{\$} [1, m]$, take $g^a \equiv X \xleftarrow{\$} \mathbb{G}$

Diffie-Hellman

Want:

① A & B get same key

② Eve can't distinguish key from unif. rand

DH Key Exchange

Fix cyclic group $\mathbb{G}$, gen. g, ord. m

Alice		Bob
$a \xleftarrow{\$} [1, m]$		$b \xleftarrow{\$} [1, m]$
$A = g^a$	$\xrightarrow{A}$	$B = g^b$
$K = B^a = g^{ab}$	$\xleftarrow{B}$	$K = A^b = g^{ab}$
ret K		ret K

- prev. our key was a bitstring... how do we use this?

+ OTP over cyclic group $\mathbb{G}$

- $\mathbb{G} = \mathcal{K}, \mathcal{M}, \mathcal{C}$

  $\underline{Enc(K, M)}$: $K \cdot M$

  $\underline{Dec(K, M)}$: $K^{-1} \cdot C$

Claim. $\forall M = g^a, K \xleftarrow{\$} \mathcal{K}, P[K \cdot g^a = g^c] = \frac{1}{m} = \frac{1}{|\mathbb{G}|}$

Pf.

$LHS = P[K = g^c \cdot g^{-a}] = P[K = g^{c-a}] = \frac{1}{|\mathbb{G}|}$ (this implies perf OTS)

- security of OHKE $\Leftrightarrow$ DDH assumption for $\mathbb{G}$

DDH Assumption. $\mathcal{L}_{DDH\text{-}real} \approx \mathcal{L}_{DDH\text{-}rand}$

$\mathcal{L}_{ddh\text{-}real}$
$a \xleftarrow{\$} \mathbb{Z}_m$
$b \xleftarrow{\$} \mathbb{Z}_m$
DDH.Get():
ret (g^a, g^b, g^{ab})

$\approx$

$\mathcal{L}_{ddh\text{-}rand}$
$a \xleftarrow{\$} \mathbb{Z}_m$
$b \xleftarrow{\$} \mathbb{Z}_m$
$c \xleftarrow{\$} \mathbb{Z}_m$
DDH.Get():
ret (g^a, g^b, g^c)

★ Adversary knows $\mathbb{G}$, g, m ★

↳ b/c these crypto schemes are public!

b/c $2 \mid |G|$, i.e., $2 \mid (p-1)$;
small prime divides $|G|$

- Is DDH assumption true?
  - ↳ $\mathbb{Z}_p^*$ (p some λ-bit prime): given $X = g^a$, can tell if a even or odd, and can thus dist. from rand (K not equally likely even/odd)

- further, discrete log hard $\not\Rightarrow$ DDH assump. true
  - ↳ e.g. $\mathbb{Z}_p^*$

Claim: DDH assump. true $\Rightarrow$ DLOG hard

Pf:

Contrapositive: assume $\exists$ poly-time DLOG solver B.

$\mathcal{A}$

$(x, y, z) = \text{DDH.Get}()$

$a, b, c = B(x), B(y), B(z)$

ret $a \cdot b \overset{?}{=}_m c$

$P[\mathcal{A} \diamond \mathcal{L}_{\text{DDH-real}} \Rightarrow T] = 1$

$P[\mathcal{A} \diamond \mathcal{L}_{\text{DDH-rand}} \Rightarrow T] = \frac{1}{m}$

$\Rightarrow DA = 1 - \frac{1}{m}$, non-negl.

Candidate DDH Groups

+ (some) cyclic groups of prime order
  - ↳ suppose p, q primes s.t. $p = 2q + 1$ (∞-many such pairs)
  - ↳ $QR_p^* = \{x^2 : x \in \mathbb{Z}_p^*\}$
    - ↳ quadratic residues mod p

E.g.

$\mathbb{Z}_{11}^* \to QR_{11}^* = \{4, 5, 9, 3, 1\}$ ($4 = 2^2$, $5 = 4^2$, $9 = 3^2$, $3 = 5^2$)

↳ assoc. ✓

↳ identity: $1 = g^{p-1} = (g^{\frac{p-1}{2}})^2$ ✓

↳ closed: $g^{2k} \cdot g^{2x} = g^{2(k+x)}$ ✓

↳ inverse: $g^{-2k} = g^{(p-1)-2k} \Rightarrow$ ✓

↳ even − even ⇒ even

Order of QR_p^* is $\frac{p-1}{2}$

QR_{11}, $g = 4$

A: $a = 2$ B: $b = 3$

↳ $A = 4^2 = 16 = 5$ $\quad 11 \cdot 5 = 55$

$B = 4^3 = 64 \cong_{11} 9$

$K = g^{ab} = 4^{3 \cdot 2} = 4^6 \cong_{11} 4$

Protocols

Def. A 2-party protocol defines a pair of algos P_1, P_2 that can send msg.s to each other, each producing an output at the end w/ all msg.s making up the transcript

↳ $(T, Y_1, Y_2) \triangleq (P_1 \rightleftarrows P_2)$

↳ e.g. in DHKE, $Y_1 = B^a$, $Y_2 = A^b$, $T = (A, B)$

Def: Key-exchange protocol.

① Correctness: $Y_1 = Y_2$, always

② $(T, K_1) \approx (T, K')$, $K' \leftarrow \mathcal{K}$

Public-Key Encryption

- Bob publishes public key Pk (Enc), has secret key Sk (Dec)

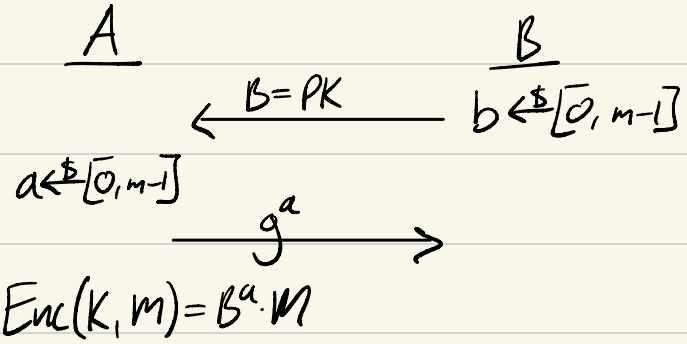
Syntax: public-key encryption scheme has 3 algos:

① KeyGen: returns (PK, SK)

② $Enc(PK, M) \to C$

③ $Dec(SK, C) \to M$

+DHKE as PKE:



$Enc(K, M) = B^a \cdot M$

"El-Gamal" enc. scheme

- i.e., $SK = b$, $PK = g^b$

$Enc(PK, M) = a \leftarrow\$ [0, m-1]$

ret $(g^a, (PK)^a \cdot M)$

8A - 2/24

PKE

- we will not discuss how we trust online public keys
  ↳ "public key infrastructure"

Def: PKE Syntax. PKE Σ has 3 algos:
- $KeyGen() \to (PK, SK)$ (blk PK & SK must be correlated)
- $Enc(PK, m) \to C$ (randomized)
- $Dec(SK, C) \to m$ (deterministic)

Correctness: $\forall\ m,\ (PK, SK) \sim KeyGen(),\ Pr[Dec(SK, Enc(PK, m)) = m] = 1$

Def: PKE CPA-security.

$\mathcal{L}_{pk\text{-}cpa\text{-}real}$

```
(PK, SK) = KeyGen()
CPA.PK():
  return PK
CPA.Enc(m):
  C = Σ.Enc(PK, m)
  ret C
```

$\approx$

$\mathcal{L}_{pk\text{-}cpa\text{-}real}$

```
(PK, SK) = KeyGen()
CPA.PK():
  return PK
CPA.Enc(m):
  C ←$ Σ.C(|m|)
  ret C
```

- recall when defining CCA sec: if you ask a box a Q, and you adv. know the answer, then the box can't help you
  ↳ in PKE, the adv. can see however many ctxts it wants!

↳ PKE: OTS ⇒ many-time security

Def: PKE 1-CPA-Security

$\mathcal{L}_{pk\text{-}1\text{-}cpa\text{-}real}$

$(PK, SK) = KeyGen()$

CPA.PK():
  return PK

CPA.Enc(m):
  if C* null:
    $C^* = \Sigma.Enc(PK, m)$
  ret C*

$\approx$

$\mathcal{L}_{pk\text{-}1\text{-}cpa\text{-}rand}$

$(PK, SK) = KeyGen()$

CPA.PK():
  return PK

CPA.Enc(m):
  if C* null:
    $C^* \xleftarrow{\$} \Sigma.\mathcal{C}(|m|)$
  ret C*

Claim. $\mathcal{L}^{\Sigma}_{pk\text{-}1\text{-}cpa\text{-}real} \approx \mathcal{L}^{\Sigma}_{pk\text{-}1\text{-}cpa\text{-}rand} \Rightarrow \mathcal{L}^{\Sigma}_{pk\text{-}cpa\text{-}real} \approx \mathcal{L}^{\Sigma}_{pk\text{-}cpa\text{-}rand}$

Pf. $(\mathcal{L}_{pk\text{-}1\text{-}cpa\text{-}real} \equiv \mathcal{L}_0 \approx \dots \approx \mathcal{L}_h \approx \mathcal{L}_{h+1} \approx \dots \approx \mathcal{L}_q \equiv \mathcal{L}_{pk\text{-}cpa\text{-}rand})$

$\mathcal{L}_h$

$(PK, SK) = KeyGen()$
CPA.PK(): ret PK
CPA.Enc(m):
  count += 1
  if count < h:
    $C \xleftarrow{\$} \Sigma.\mathcal{C}(|m|)$
  else:
    $C = \Sigma.Enc(PK, m)$
  ret C

$\equiv$

$\mathcal{L}_h$

$(PK, SK) = KeyGen()$
CPA.PK(): ret PK
CPA.Enc(m):
  count += 1
  if count < h:
    $C \xleftarrow{\$} \Sigma.\mathcal{C}(|m|)$
  if count = h:
    $C = \Sigma.Enc(PK, m)$
  else:
    $C = \Sigma.Enc(PK, m)$
  ret C

$\equiv$

$\approx$

$\mathcal{L}_h$

$PK = 1\text{-}CPA.PK()$
CPA.PK(): ret PK
CPA.Enc(m):
  count += 1
  if count < h:
    $C \xleftarrow{\$} \Sigma.\mathcal{C}(|m|)$
  if count = h:
    C = CPA.Enc(m)
  else:
    $C = \Sigma.Enc(PK, m)$
  ret C

◇ $\mathcal{L}_{pk\text{-}1\text{-}cpa\text{-}real}$

◇ $\mathcal{L}_{pk\text{-}1\text{-}cpa\text{-}rand}$ by the 1-CPA-sec of Σ

then just pull external library back in, and note $q = poly(\lambda) \Rightarrow$ valid hybrid argument.

Also note: $\mathcal{L}_{pk\text{-}cpa\text{-}1\text{-}real} \equiv \mathcal{L}_0$

$\mathcal{L}_q \equiv \mathcal{L}_{pk\text{-}cpa\text{-}rand}$

El Gamal

$\mathcal{M} = \mathbb{G}$, $\mathcal{C} = \mathbb{G} \times \mathbb{G}$, $PK \in \mathbb{G}$, $SK \in \mathbb{Z}_{|\mathbb{G}|}$

KeyGen():
- $SK \xleftarrow{\$} \mathbb{Z}_m$
- $PK = g^{SK}$
- ret (PK, SK)

Enc(PK, M):
- $r \xleftarrow{\$} \mathbb{Z}_m$
- $C_1 = g^r$
- $C_2 = M \cdot PK^r$
- ret (C_1, C_2)

Dec(SK, (C_1, C_2)):
- $M = C_2 \cdot C_1^{-SK}$
- ret M

Claim. DDH assumption for $\mathbb{G} \Rightarrow$ El Gamal CPA-security.

Pf.

$\mathcal{L}_{pk\text{-}1\text{-}cpa\text{-}real}$

- $SK \xleftarrow{\$} \mathbb{Z}_m$
- $PK = g^a$
- 1-CPA.Enc(M):
  - if C_1^* null:
    - $r \xleftarrow{\$} \mathbb{Z}_m$
    - $C_1^* = g^r$
    - $C_2^* = M \cdot PK^r$
  - ret C_1^*, C_2^*

$\equiv$

$\mathcal{L}_{pk\text{-}1\text{-}cpa\text{-}real}$

- $SK \xleftarrow{\$} \mathbb{Z}_m$
- $PK = g^a$
- $r \xleftarrow{\$} \mathbb{Z}_m$
- $C_1^* = g^r$
- $U = PK^r = (g^{SK})^r$

g^a, g^b, g^{ab} from DDH! ↳ 3-hop them out

- 1-CPA.Enc(M):
  - if (C_1^*, C_2^*) null:
    - $C_2^* = M \cdot U$
  - ret (C_1^*, C_2^*)

8B - 2/26

- in CCA, integrity was important
  - ↳ MACs
    - ↳ in PKE called digital signature

Watermarking for LLMs

- goal: distinguish LLM-generated from human text
  - ↳ new idea in past 3 years!

Notations

- next-token predictor M, vocabulary T
- input prompt $P = P_1 P_2 \dots P_k \in T^*$
  - ↳ output of M is dist. over next $t \in T$
- we can define a model:

$\overline{M}(P)$:

```
T = empty string
while True:
  D = M(P||T)
  t ← sample from D
  T = T||t
  if t = ⊥:
    ret T
```

Watermarking

- 2 algos:

① Mark: generates watermarked text, using M

② Detect: given input, returns True (AIG) or False (not AIG)

↳ both use secret key K

Formally:

- $Mark(K, P) = T \in T^*$
- $Detect(K, T) = B \in \{True, False\}$

Goals

- marked model should be a good model (lol) → quality
- hard to get rid of watermark; hard to create text that's "close to marked" → robustness
- $Detect(K, Mark(K, P)) = True$ → correctness

  $Detect(K, non\text{-}Mark) = False$ → low FPR (soundness)

"Give me a name for a rabbit. Sacramento."

Def: Correctness. $Pr[Detect(K, Mark(K, P)), H_m(P, T)\ high] = negl.$

↳ $H_m(P, T) = -\log(Pr[T|P])$ (entropy)

↳ for robustness, add a line that "$T' \approx Mark(K, P)$" & detect on T'

Def: Low False Positives.

$\mathcal{L}^{W}_{\text{fp-real}}$	$\approx$	$\mathcal{L}^{W}_{\text{fp-ideal}}$
$K \xleftarrow{\$} \mathcal{K}$ FP.Detect(T): ret Detect(K, T)		FP.Detect(T): ret False

Def: Undetectability.

$\mathcal{L}_{\text{und-real}}$	$\approx$	$\mathcal{L}_{\text{und-ideal}}$
$K \xleftarrow{\$} \mathcal{K}$ UND.Generate(P): ret Mark(K, P)		UND.Generate(P): ret $\overline{M}(P)$

The Attack

- given a prompt P, ask the marked model many queries $P_1, P_2, \ldots$ get responses $T_1, T_2, \ldots$, and return some T
  - ↳ this is equivalent to $P \to \boxed{\overline{M}} \to T$!

Thm. Suppose W is an undetectable wm scheme. Then, for any prompt P, $\exists$ adv. $\mathcal{A}$ s.t. $\forall z \in \Upsilon^*$,

$$\left| \Pr[\mathcal{A} \diamond \mathcal{L}_{\text{und-real}} \Rightarrow z] - \Pr[\overline{M}(P) = z] \right| = \text{negl}.$$

$\mathcal{A}$

```
T = empty string
while True:
  T' = Und.Generate(P||T)
  t = T'[0]
  T = T||t
  if t = ⊥:
    ret T
```

+ proof outline: $\mathcal{L}_{\text{und-real}}$ & $\mathcal{L}_{\text{und-ideal}}$'s dist over the next token must be indistinguishable, making $\mathcal{A}$ equiv. to $\overline{M}$
  ↳ if they weren't indistinguishable, W wouldn't be undetectable

9A - 3/3

Back to Watermarking

- M is a NTP, fixed prompt $P \in \{0,1\}^*$
  - ↳ t_i denotes the ith bit of T

- recall our generate function:

$\overline{M}.\text{Generate}(P)$:

```
T = ε  // empty
while ⊥ ∉ T:
    p = M(P || T)
    t ~ Bern(p)
    T = T || t
ret T
```

how do we sample?

Sampling $t \sim \text{Bern}(p)$

```
r ←$ [0, 1]
if r ≤ p:
    t = 1
else:
    t = 0
```

- we'll interp. $F(K, X) \to \{0,1\}^m$ as an int $\in \{0, 2^m - 1\}$
  - ↳ then $\frac{F(K,X)}{2^m - 1} \in [0, 1]$

- note that if we keep our values from Sample = $R = r_1 r_2 \dots r_n$, these correlate w/ $T = t_1 t_2 \dots t_n$

Claim. $\exists$ a fn. Corr s.t. for any P, w/ $R \in [0,1]^n$ the randomness used to sample T, if $H_\infty(P, T)$ large enough,

$$\Pr[\text{Corr}(R, T) = \text{false}] \leq \text{negl}(\lambda)$$

Moreover, $\forall Y \in [0,1]^n$ ind. of T, $\Pr[\text{Corr}(R, T) = \text{true}] \leq \text{negl}(\lambda)$.

Constructing W

+ Attempt #1 (doesn't work!)

Mark(K, P):

```
T = ε
i = 1
while ⊥ ∉ T:
    p_i = M(P||T)
    r_i = F(K, i)
    if r_i ≤ p_i:
        t_i = 1
    else:
        t_i = 0
    T = T||t_i
    i += 1
ret T
```

→ problem: this is deterministic
↳ "not random enough" for PRF security

+ The fix:

Mark(K, P):

```
T = ε, i = 1, H = 0
while ⊥ ∉ T:
    p_i = M(P||T)
    if H < λ:
        r_i ←$ [0, 1]
        if r_i ≤ p_i : t_i = 1
        else t_i = 0
        H = H - log p_i(t_i)
    elif H ≥ λ:
        τ = t_1 t_2 ... t_i
        r_i = F(K, τ||i)
        if r_i ≤ p_i : t_i = 1
        else: t_i = 0
    T = T||t_i
    i = i+1
```

```
Detect(K, T):
  for i ∈ {1, 2, ..., |T|}
    τ = t_1 ... t_i
    T' = t_{i+1} ... t_{|T|}
    R = ε
    for j ∈ {1, ..., |T'|}:
      r_i = F(K, τ||_j)
      if Corr(R, T'): ret True
  ret False
```

Proofs

Claim. W is undetectable.

Pf.

$\mathcal{L}_{\text{und-real}}$	$\equiv$	
⋮		
if $H \geq \lambda$		if $H \geq \lambda$:
$\tau = t_1 \ldots t_i$		$\tau = t_1 \ldots t_i$
		if $L[\tau]$ null:
		$L[\tau] = \tau$
		else:
		BAD = True
		ret $\perp$

But $H \geq \lambda \Leftrightarrow \Pr[\text{we've seen } \tau] \leq 2^{-\lambda}$. And w/ negl. probability of colliding inputs, we can replace $F(K, \cdot)$ w/ $\xleftarrow{\$} [0,1]$ by F's security. Then the proof is done.

Thm. W has low FPs.

Pf.

$\mathcal{A}$ submits $T_1, \ldots, T_q$, $q = \text{poly}(\lambda)$. Then

$\mathcal{L}^h$	$\overset{?}{\approx}$	$\mathcal{L}^{h+1}$
$K \xleftarrow{\$} W.\mathcal{K}$		$K \xleftarrow{\$} W.\mathcal{K}$
FP.Detect(T_i)		FP.Detect(T_i)
if $i \le h$:		if $i \le h+1$:
ret False		ret False
else:		else:
$B = W.\text{Detect}(K, T_i)$		$B = W.\text{Detect}(K, T_i)$
ret B		ret B

$\mathcal{L}^h$ and $\mathcal{L}^{h+1}$ could only differ on the $(h+1)$th query, T_{h+1}. The proof follows from the fact that b/c T_{h+1} is ind. of $F(K, \cdot)$, and b/c $\tau \| j$ is always unique, $r_i = F(K, \tau \| j)$ can be replaced with $r_i \xleftarrow{\$} [0,1]$. Then $\text{Corr}(Y, T')$ must be negl., so $\mathcal{L}^h$ will ret. true negligibly. $\Rightarrow \mathcal{L}^h \approx \mathcal{L}^{h+1}$

And trivially real $\equiv \mathcal{L}^0$, $\mathcal{L}^q \equiv$ ideal.

Union bound then proves real $\approx$ ideal.

Thm. W is correct.

Pf.

Fix a $K \in W.\mathcal{K}$, $P \in \{0,1\}^*$, $T = W.\text{Mark}(K, P)$. Suppose $H_m(P, T)$ is large enough.
Then, by the Claim, Detect will output true whp.

Robustness of Watermarks

- correctness doesn't tell us anything a/b if we edit the wm
  - ↳ this scheme is robust to cutting off the end of the T, but not the T